Using Weak Supervision to Identify Long-Tail Entities for Knowledge Base Completion

Yaser Oulabi(✉) and Christian Bizer

Data and Web Science Group, University of Mannheim,
B6 26, 68159 Mannheim, Germany
{yaser,chris}@informatik.uni-mannheim.de

Abstract. Data from relational web tables can be used to augment cross-domain knowledge bases like DBpedia, Wikidata, or the Google Knowledge Graph with descriptions of entities that are not yet part of the knowledge base. Such long-tail entities can include for instance small villages, niche songs, or athletes that play in lower-level leagues. In previous work, we have presented an approach to successfully assemble descriptions of long-tail entities from relational HTML tables using supervised matching methods and manually labeled training data in the form of positive and negative entity matches. Manually labeling training data is a laborious task given knowledge bases covering many different classes. In this work, we investigate reducing the labeling effort for the task of long-tail entity extraction by using weak supervision. We present a bootstrapping approach that requires domain experts to provide a small set of simple, class-specific matching rules, instead of requiring them to label a large set of entity matches, thereby reducing the human supervision effort considerably. We evaluate this weak supervision approach and find that it performs only slightly worse compared to methods that rely on large sets of manually labeled entity matches.

1 Introduction

Cross-domain knowledge bases like YAGO [8], DBpedia [9], Wikidata [20], or the Google Knowledge Graph are being employed for an increasing range of applications, including natural language processing, web search, and question answering. The entity coverage of knowledge bases is far from complete [4,16]. YAGO and DBpedia e.g. rely on data extracted from Wikipedia and as a result cover mostly head instances that fulfill the Wikipedia notability criteria [12]. As the utility of a knowledge base increases for many tasks with its completeness, adding long-tail entities to a knowledge base is an important task.

Web tables [3], which are relational HTML tables extracted from the Web, contain large amounts of structured information, covering a wide range of topics. In previous work [12], we proposed a method for extracting long-tail entities and showed that web tables are a promising source for augmenting knowledge bases

M. Acosta et al. (Eds.): SEMANTiCS 2019, LNCS 11702, pp. 83–98, 2019.
https://doi.org/10.1007/978-3-030-33220-4_7

with new and formerly unknown entities. For this, we trained models using large sets of manually labeled class-specific entity matches. Given that knowledge bases can have many classes, manual labeling limits the usefulness of automatic knowledge base augmentation from web tables.

Weak supervision approaches aim at reducing labeling effort by using supervision that is more abstract or noisier compared to traditional manually labeled high-quality training examples (strong supervision) [14]. Data programming [15] is a paradigm, where experts are tasked with codifying any form of weak supervision into labeling functions. These functions are then employed within a broader system to generate training data by assigning labels and confidence scores to unlabeled data. Recently, various different systems based on the data programming paradigm have been suggested [1,14,19].

For many types of entities, humans generally possess knowledge about when entities definitely match, and what are strong signals that entities do not match. Writing down this general knowledge in the form of simple bold matching rules requires far less effort than labeling many individual positive and negative entity matches. Building on this observation and the data programming paradigm, this paper investigates for the task of long-tail entity extraction whether strong supervision in the form of positive and negative entity matches can be replaced by a set of simple bold matching rules. In order to make it easy to write down such rules, we restrict the rule format to conjuncts of equality tests. These tests are expressed using the schema of the knowledge base without requiring experts to assign weights or specify similarity metrics. Additionally, we introduce a bootstrapping method that exploits the matching rule sets to generate training data and train a supervised machine learning algorithm. Using these approaches, we are able to significantly reduce supervision effort compared to manually labeling positive and negative entity matches, while achieving a comparable performance.

Our contributions are (1) a weak supervision approach that substitutes manually labeled training pairs by a set of bold matching rules, (2) a bootstrapping approach which uses weak supervision to generate training data for a supervised matching method, and (3) an evaluation that compares strong and weak supervision for the task of long-tail entity extraction.

The remainder of this paper is structured as follows. First, we describe our long-tail entity extraction method, including the experimental setup and a summary of results when using strong supervision. Section 3 describes our weak supervision methodology, while Sects. 4 and 5 present and discuss our experiments. Section 6 compares our approach to the related work. The results presented in this paper are fully reproducible, as we publicly provide all code and datasets.[1]

2 Long-Tail Entity Extraction

In previous work, we proposed and evaluated a method for long-tail entity extraction from web tables [12]. This section summarizes the proposed approach, describes our experimental setup, and presents results achieved using manually labeled training data.

[1] http://data.dws.informatik.uni-mannheim.de/expansion/LTEE/.

2.1 Methodology

Extracting long-tail entities from web tables for knowledge base augmentation is a non-trivial task. It consists of two subtasks: (1) identifying entities that are not yet part of the knowledge base and (2) compiling descriptions for those new entities from web table data according to the schema of the knowledge base.

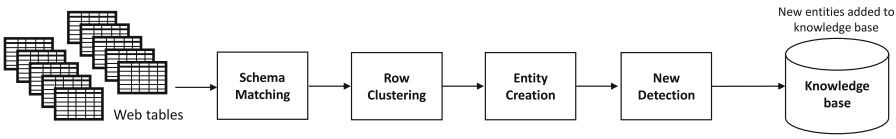

Fig. 1. Pipeline for extending a knowledge base with long-tail entities from web tables.

Long-Tail Entity Extraction Pipeline. Figure 1 gives an overview of our suggested approach. It is a pipeline that starts with web tables and ends by adding new entities to a cross-domain knowledge base. We first cluster all rows that describe the same real-world instance together. From these clusters we then create entities by compiling descriptions from web table data. Finally, the new detection component determines which entities are new, given a specific target knowledge base. As a result, we are able to perform the two subtasks of identifying new entities and compiling their descriptions.

Schema Matching. The first component of the pipeline is schema matching. It creates a mapping between web tables and the knowledge base schema. This includes matching web tables to classes and web table columns to properties. The latter, termed attribute-to-property correspondences [17], allow us to semantically understand cell values. They are exploited by the entity creation component to compile description according to the schema of the knowledge base and by both, the row clustering and new detection components, as similarity features.

Performing Row Clustering and New Detection. For both, row clustering and new detection, we train random forest classifiers that perform entity matching. For row clustering, the classifier compares a row pair to determine if the two rows describe the same entity, while for new detection this is done for a pair of a created entity and a candidate instance from the knowledge base.

Comparing all possible row pairs or entity-instance-pairs would not scale. We therefore utilize a label-based blocking approach using a Lucene index to find candidates to be compared.

Each matching decision is also given a confidence score. For row clustering, we use the confidence scores to perform correlation clustering and generate the row clusters. For new detection, we return an entity as new, only if all candidate instances from the knowledge base were classified as clear non-matches.

Table 1. Overview of the number of labels in the T4LTE gold standard.

Label type	GF-Player	Song	Settlement	Sum
Row pair	1,298	231	2,768	**4,297**
Entity-instance-pair	80	34	51	**165**
New entity classification	17	63	23	**103**
Sum	**1,395**	**328**	**2,842**	**4,565**

Similarity Features. To train a classifier, we exploit various features, which are described in more details in our previous work [12]. Among the features are first the similarities of labels (`LABEL`) and bag-of-words vectors (`BOW`). Secondly, using the attribute-to-property correspondences we derive values according to the knowledge base schema, which we compare using data-type-specific similarity functions (`ATTRIBUTE`). Using the knowledge base we also derive for each table implicit attributes about the entities described in the table, giving us another set of values by knowledge base property that we compare using data-type-specific similarity functions (`IMPLICIT_ATT`). For row clustering, we additionally exploit the PHI correlation of row labels (`PHI`) and penalize rows which occur in the same table (`SAME_TABLE`). For new detection, we additionally exploit type overlap between a created entity and a candidate knowledge base instance (`TYPE`), and the popularity of a candidate knowledge base instance (`POPULARITY`).

For each row pair or entity-instance-pair most features return a single normalized similarity score. For `ATTRIBUTE` and `IMPLICIT_ATT`, we return for a pair two scores for each property from the knowledge base schema. One score measures the confidence of the pair having equal values given that property, the other of the pair having unequal values.

2.2 Experimental Setup and Results

We employ the 2014 release of DBpedia [9] as the target knowledge base and evaluate our methods on the task of extending the DBpedia classes GridironFootballPlayer (GF-Player), Song[2], and Settlement with additional entities. To ensure diversity among the classes, we selected each from a different first-level class, i.e. Agent, Work, and Place.

We utilize the English-language relational tables set of the Web Data Commons 2012 Web Table Corpus.[3] The set consists of 91.8 million tables. For every table we assume that there is one column that contains the labels of the instances described by the rows. The remaining columns contain values, which potentially can be matched to properties in the knowledge base schema.

For training and evaluation we built the *Web Tables for Long-Tail Entity Extraction*[4] (T4LTE) gold standard. Table 1 provides an overview of the number of labels in T4LTE. Creating this dataset was rather laborious, as we

[2] The class Song also includes all instances of the class Single.
[3] http://webdatacommons.org/webtables/#toc3.
[4] http://webdatacommons.org/T4LTE/.

labeled 4,297 matching row pairs, 165 entity-instance-pairs and 103 new entity classifications.

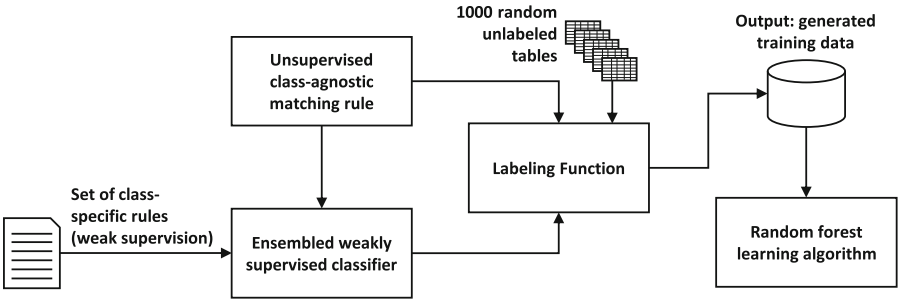

Fig. 2. Our overall methodology of introducing weak supervision using class-specific rule sets and bootstrapping a supervised learning algorithm using a labeling function.

When evaluating the pipeline using the T4LTE gold standard using cross-validation, we were able to achieve an F1 score in the task of finding new entities of 0.80. When running the pipeline on the whole web table corpus, we were able to add 14 thousand new gridiron football players and 187 thousand new songs to DBpedia, an increase of 67% and 356% respectively [12].

3 Methodology

This section describes our approaches for the task of reducing labeling effort using weak supervision. The overall methodology is illustrated in Fig. 2.

We first introduce as a baseline two unsupervised class-agnostic matching rules for row clustering and new detection. These rules exploit the similarity features described above and aggregate them using a weighted average.

We then introduce an approach that exploits user-provided class-specific rule sets as weak supervision. These rules have a high accuracy, but low coverage, which is why we ensemble them with the unsupervised matching rule to derive weakly supervised classifiers for both row clustering and new detection.

Both, the unsupervised matching rules and the weakly supervised classifiers can be used in our pipeline directly. We additionally introduce an approach that exploits these methods as labeling functions to bootstrap a supervised learning algorithm. This is done by using a set of unlabeled web tables to label training pairs for both row clustering and new detection. The labeled data is then used to train random forest classifiers to be used in our pipeline.

3.1 Unsupervised Class-Agnostic Matching Rule

We suggest two unsupervised matching rules that aggregate using a weighted average the individual scores generated by the features described in Sect. 2.1.

To be used in a rule, all features must produce scores that are normalized and class-agnostic. This already applies to all features except `ATTRIBUTE` and `IMPLICIT_ATT`, where, given a pair, we normalize by averaging the individual property scores, giving us one normalized class-agnostic score per feature.

We determine the weights of the rules by assigning, based on our own experience with the metrics, importance factors from 4 to 1 to the individual features. The weight of a feature is equal to it assigned factor normalized by the sum of all factors. For the row clustering rule we assign a factor of 4 to `LABEL`, 2 to `BOW` and `ATTRIBUTE`, and 1 to `PHI`, `IMPLICIT_ATT` and `SAME_TABLE`. For new detection we assign a factor of 4 to `LABEL`, 3 for `BOW` and `ATTRIBUTE`, 2 for `TYPE` and `IMPLICIT_ATT`, and 1 for `POPULARITY`.

The rules determine whether a pair matches or not using a fixed threshold, simply set at 0.5 for both rules. The absolute distance of a computed average from the threshold determines the confidence of a matching decision.

3.2 Class-Specific User-Provided Matching Rules

Humans often possess general knowledge about which conditions need to be fulfilled for entities of a certain domain to clearly match or clearly not match. Based on this observation, we suggest as weak supervision a set of user-provided bold class-specific rules that classify a given candidate pair as a match or non-match. They can codify obvious knowledge, e.g. that a settlement can not be in two different countries, or non-obvious knowledge, e.g. that only one unique football athlete can be drafted in the same year with the same pick number.

The rules consists of conjunctions of attribute tests, expressed using the schema of the knowledge base. It is only required that the provided rules be accurate, regardless of their coverage. This makes it a simple task to identify suitable rules and is the reason why we term these rules as bold. For our experiments, we created per class four rules. For **GF-Player** we came up with two matching and two non-matching rules:

$$(\text{draftYear} = \text{Equal}) \wedge (\text{draftPick} = \text{Equal}) \rightarrow \text{Match} \tag{1}$$

$$(\text{LABEL} = \text{Equal}) \wedge (\text{birthDate} = \text{Equal}) \rightarrow \text{Match} \tag{2}$$

$$(\text{draftYear} = \text{Unequal}) \rightarrow \text{Non-Match} \tag{3}$$

$$(\text{draftPick} = \text{Unequal}) \rightarrow \text{Non-Match} \tag{4}$$

For **Song** we also came up with two matching and two non-matching rules:

$$(\text{LABEL} = \text{Equal}) \wedge (\text{artist} = \text{Equal}) \wedge (\text{releaseDate} = \text{Equal}) \rightarrow \text{Match} \tag{5}$$

$$(\text{LABEL} = \text{Equal}) \wedge (\text{artist} = \text{Equal}) \wedge (\text{album} = \text{Equal}) \rightarrow \text{Match} \tag{6}$$

$$(\text{artist} = \text{Unequal}) \rightarrow \text{Non-Match} \tag{7}$$

$$(\text{releaseYear} = \text{Unequal}) \rightarrow \text{Non-Match} \tag{8}$$

Finally, for **Settlement** we have three matching and one non-matching rule:

$$(\text{country} = \text{Equal}) \wedge (\text{postalCode} = \text{Equal}) \rightarrow \text{Match} \tag{9}$$

$$(\text{LABEL} = \text{Equal}) \wedge (\text{isPartOf} = \text{Equal}) \rightarrow \text{Match} \tag{10}$$

$$(\text{LABEL} = \text{Equal}) \wedge (\text{postalCode} = \text{Equal}) \rightarrow \text{Match} \tag{11}$$

$$(\text{country} = \text{Unequal}) \rightarrow \text{Non-Match} \tag{12}$$

The effort spent creating these rules is minuscule compared to manually labeling the correspondences in the gold standard. While for each class we created only 4 rules, they are tested to substitute 1,395, 328, and 2,842 labels for the classes GF-Player, Song, and Settlement respectively.

To apply a rule we exploit the equal and unequal scores generated by the `ATTRIBUTE` and `IMPLICIT_ATT` features, as described in Sect. 2.1, and the `LABEL` feature using a data-type specific equivalence threshold [12]. A rule fires, when all tests within the rule have scores higher than zero. From these scores we also derive for each rule firing a confidence score, which equals the product of all scores used within the rule.

As the rules fire only when certain conditions are met, the set of rules is not exhaustive and only covers a subset of compared pairs. We therefore ensemble the rules with the unsupervised matching rule through averaging. Given a compared pair, we first check how many rules fire. If no rule fires, we simply return the output of the unsupervised matching rule. If multiple rules fire, which is possible as the rules are not mutually exclusive, we consider only the rule with the highest confidence, preferring negative rules in case of a tie. If the confidence of this rule is higher than the confidence of the output of the unsupervised matching rule, the outputs of both are averaged and returned. Otherwise, we simply return the output of the unsupervised matching rule.

3.3 Bootstrapping Approach

In our experiments, we, on the one hand, directly apply the unsupervised rule and the weakly supervised ensembled classifier to our test data. On the other hand, following the data-programming paradigm, we employ both methods as labeling functions to label row pairs and entity-instance-pairs derived from 1000 randomly selected web tables as matches or non-matches. Additionally, the labeling functions assign weights to the training examples using the confidence scores returned by the underlying method. Using these labels we train a random forest classifier, which is then applied to our test data.

To derive pairs to be labeled, we employ label-based blocking using Lucene for both row clustering and new detection. We additionally include random pairs to be labeled, for row clustering as many as there are positive pairs, and for new detection 8 random instances selected from the knowledge base from within the same class of an entity or its parent classes. Overall, this leads to 2.8 m row pairs and 1.27 m entity-instance-pairs selected to be labeled.

For row clustering, we use the confidence scores to additionally perform correlation clustering. A row pair labeled as a match but not part of the same cluster, is not included as a positive training example. Similarly, a row pair labeled as a non-match, but placed in the same cluster, is not considered as a negative training example.

For new detection, when multiple entity-instance-pairs of the same entity are labeled as matching, which can not be correct, we only include the entity-instance-pair with the highest score as a positive training example.

Table 2. Row clustering performance for runs with various types of supervision.

Method	Average			GF-Player	Song	Settlement
	PCP	AR	F1	F1	F1	F1
Unsupervised	0.76	0.86	0.80	0.90	0.65	0.86
+ Bootstrapping	0.78	0.88	0.83	0.89	0.73	0.86
Weak supervision	0.83	0.89	0.86	0.93	0.81	0.84
+ Bootstrapping	0.83	0.90	0.86	0.89	0.83	0.86
Strong supervision	0.86	0.90	0.88	0.91	0.84	0.90
+ Bootstrapping	0.85	0.90	0.87	0.92	0.79	0.91

When bootstrapping for new detection, we also need a set of row clusters from which we create entities. Using these entities we can then generate training examples using entity-instance-pairs and our labeling function. To create these clusters, we use the supervised model trained by bootstrapping from a labeling function of equal supervision, i.e. when we are bootstrapping a supervised learning algorithm for new detection using the unsupervised rule, we use the clustering method also trained using bootstrapping and the unsupervised rule.

Given the labeled pairs, we train a random forest classifier. Per forest, we train 2000 trees. To reduce correlation between trees, we set the features available at each split to 2, and reduce the sample size used to train each tree to 66% of the total number of pairs. We sample with replacement and using weights, so that higher weighted examples are considered more often during training.

4 Evaluation and Results

In this section, we evaluate, using the T4LTE gold standard, the approaches described above and compare them to a model trained with manually labeled data. As for the latter, the gold standard is also used for training, we apply three-fold cross-validation throughout all experiments. Additionally, we will evaluate the effectiveness of the user-provided rule sets and our bootstrapping approach.

4.1 Row Clustering Evaluation

To evaluate row clustering, we employ the evaluation metric proposed by Hassanzadeh et al. [7,12]. It emphasizes replicating the exact number of clusters in the evaluation set by first computing a one-to-one mapping between returned clusters and clusters in the evaluation set. Only rows of clusters with a mapping contribute towards recall, while the pairwise clustering precision is penalized by the difference between the number of clusters in the evaluation set and the number of returned clusters, or the clusters with a mapping, whichever is higher.

Table 2 shows row clustering performance for different types of supervision. The first two rows show performances when using the unsupervised matching rule alone, while the following two rows show the performances when using the weakly supervised ensembled classifier. The final two rows show the performances when using strong supervision. For each supervision type we apply and evaluate the underlying method directly on the test set, and then use it as a labeling function to bootstrap a random forest, which we then also apply and evaluate on the test set. For strong supervision, the bootstrapped method resembles a semi-supervised learning approach.

From the table, we can see that the difference in average F1 between a model trained using strong supervision, which has an F1 of 0.88, and the unsupervised rule without bootstrapping is 8 pp. We find that using bootstrapping with the unsupervised matching rule allows us to increase F1 by 3 pp on average, with an increase of 8 pp for the class Song. Using user-provided class-specific rule sets, we achieve an average F1 score of 0.86, which is a large increase of overall 6 pp from the unsupervised rule and very close to the performance when using strong supervision. Applying bootstrapping on the weakly supervised method does not increase average F1 further, mainly because we lose performance for the class GF-Player, while gaining performance in the other two classes. This is similarly the case when bootstrapping from a model trained using strong supervision, except we also lose one percentage point in average F1.

When bootstrapping, the labeling functions were given overall 2.8 m row pairs to label, which were selected either by the label-based blocker or chosen randomly. Given as labeling function the weakly supervised ensembled classifier, 275 thousand pairs were labeled as matches, while 2.54 m pairs were labeled as non-matches. For this output, the user-provided matching rules fire in total 37 thousand times, whereas the non-matching rules fire 500 thousand times.

4.2 New Detection Evaluation

We evaluate a new detection method using both, the existing and the new entities labeled in the gold standard. Precision equals the proportion of entities returned as new by the method, that are actually new, while recall equals the proportion of new entities in the testing set, that were returned as new by the method.

Table 3 shows new detection performance for runs with various types of supervision, similar to Table 2. We first find that a model trained using the provided strong supervision outperforms the unsupervised matching rule in F1 by 7 pp on

Table 3. New detection performance for runs with various types of supervision.

Method	Average			GF-Player	Song	Settlement
	P	R	F1	F1	F1	F1
Unsupervised	0.87	0.76	0.80	0.82	0.68	0.89
+ Bootstrapping	0.86	0.86	0.85	0.86	0.78	0.90
Weak supervision	0.87	0.81	0.83	0.82	0.78	0.89
+ Bootstrapping	0.87	0.90	0.87	0.87	0.85	0.90
Strong supervision	0.82	0.94	0.87	0.88	0.92	0.81
+ Bootstrapping	0.81	0.97	0.88	0.88	0.92	0.83

average, and by 24 points for the class Song. On the other hand, the unsupervised matching rule outperforms the model trained using strong supervision by 8 pp for the class Settlement, indicating that the trained model highly overfits. By employing the user-provided rule sets as weak supervision, we are able to increase average F1 by 3 pp.

Unlike for row clustering, bootstrapping is consistently effective for new detection. It increases average F1 in the unsupervised case by 5, and in the weakly supervised case by 3 pp. The latter allows us to achieve an equal average F1 to that of strong supervision, albeit a large part is due to the Settlement class, while for Song we are still lacking 7 points in F1. Bootstrapping is also effective when used with a model trained using strong supervision.

When bootstrapping, a sum of 1.27 m entity-instance-pairs are given to the labeling functions to be labeled. When using the ensembled classifier, we find that 26 thousand pairs were labeled as matches, and the remainder as non-matches. Within the ensembled classifier, the user-provided matching rules fire 13 thousand times, whereas the non-matching rules fire 150 thousand times.

4.3 End-To-End Evaluation

We will now evaluate a full run of the pipeline using weak supervision. As this runs row clustering and new detection sequentially, the errors of the methods tend to accumulate and reduce overall end-to-end performance [12].

To evaluate how well new entities were found, we utilize precision and recall. To compute precision, we determine the proportion of entities returned as new that are correct. An entity is only correctly new, if its cluster includes the majority of the rows of a new cluster in the gold standard, and these rows at the same time form the majority within the entity's cluster. Recall is the fraction of new entities in the gold standard for which a correct new entity was returned.

Table 4 shows end-to-end performance for different types of supervision similar to Table 2. From the table we can see that the highest performance is achieved by the model trained using strong supervision. It achieves an average F1 of 0.81. The highest performance achieved by the methods without strong supervision

Table 4. End-to-end evaluation for various types of supervision.

Method	Average			GF-Player	Song	Settlement
	P	R	F1	F1	F1	F1
Unsupervised	0.71	0.71	0.69	0.76	0.50	0.82
+ Bootstrapping	0.71	0.81	0.74	0.79	0.60	0.82
Weak supervision	0.72	0.77	0.74	0.76	0.63	0.82
+ Bootstrapping	0.72	0.86	0.78	0.81	0.72	0.80
Strong supervision	0.73	0.93	0.81	0.84	0.78	0.81
+ Bootstrapping	0.68	0.93	0.78	0.84	0.69	0.80

is 0.78 for the weak supervision method with bootstrapping. The lowest performance of 0.69 is achieved by the unsupervised method without bootstrapping. Overall, we find that we are able to achieve a performance quite close to that when using strong supervision, and much better than a simple unsupervised matching rule. As a result, we can successfully perform long-tail entity extraction with significantly reduced labeling effort. While on average, we lose recall with almost no loss in precision, the actual effect differs per individual class.

The user-provided rule sets have a strong positive impact on performance, increasing F1 by 5 pp. Bootstrapping also increases average F1 by 5 and 4 pp for the unsupervised and weakly supervised runs respectively. Overall, we achieve an increase of 9 points when comparing a weakly supervised bootstrapped method with an unsupervised non-bootstrapped method. The effect is especially large for Song, where we gain 22 pp in F1.

Bootstrapping from a strongly supervised method is not effective and reduces overall performance. This is because, bootstrapping had mixed results when it comes to row clustering for both, weak and strong supervision. This is especially the case for the class Song, where a method bootstrapped from strong supervision produces 29 bad clusters, leading to a significant drop in end-to-end performance.

Finally, we notice that precision is continuously lower than recall. For GF-Player and Settlement we have e.g. precisions of 0.68 and 0.70, with recalls of 1.00 and 0.92 respectively. This problem is caused by bad clustering, primarily for existing entities, which are then classified as new by the new detection component, thereby reducing precision, without affecting recall. When summing numbers for all testing folds, we are missing for football players 8 existing clusters, meaning the rows were incorrectly included in other existing clusters, causing them to be impure. In the case of settlements we have overall generated 16 extra existing clusters. This leads for GF-Player and Settlements to 8 and 9 clusters respectively, being incorrectly determined to be new. This shows, that errors in the pipeline accumulate and that there is a need for an additional component in the pipeline that detects and filters out bad clusters. While this pattern does not exist for class Song, it is because it suffers from bad clustering for new and existing clusters, leading to lower recall and precision. As a result, even the class Song would benefit from a bad cluster filtering component.

5 Discussion

Ensembling the user-provided rule sets with an unsupervised matching rule, yields a quite effective method that requires minimal supervision. The unsupervised rule, while class-agnostic and simple, still provides an acceptable baseline performance, and more importantly, full coverage to our method. This allows us to require that the rules only be accurate, but not exhaustive, even when the number of provided rules is small. Additionally, these rules are not only easily created by an expert, but could also be mined from or tested on the knowledge base, further reducing supervision effort. A big limitation of our approach is that the rule sets require web tables to describe entities using useful knowledge base properties. This is not the case for settlements, where we find that the number and density of attributes in the web tables are limited [12].

While bootstrapping produces mixed results for row clustering, its impact on new detection and end-to-end performance is positive. There are several factors that possibly contribute to this positive effect. First, a random forest is more expressive than either, the unsupervised matching rule or the user-provided rule sets. It also exploits a larger feature set than both, especially making use of the class-specific scores returned by the `ATTRIBUTE` and `IMPLICIT_ATT` features. By weighting training pairs, we ensure that pairs with a higher confidence are given a higher importance, while less certain pairs are still considered. As bootstrapping works within the context of a component, i.e. row clustering or new detection, it can make use of component-specific characteristics. For example, given one created entity, only one knowledge base instance can possibly be a correct match. This allows us to eliminate likely incorrect training examples during bootstrapping for new detection by keeping for one entity only the matching entity-instance-pair with the highest confidence.

6 Related Work

Various methods exist to reduce effort spent on manual labeling. Semi-supervised methods use a small set of labeled and a larger set of unlabeled examples to train a model. This includes for example co-training and self-training, which train models on data that they labeled themselves, using initially a small number of seed examples. Another approach to reducing labeling effort is active learning, where a user is queried to label examples that are chosen to provide the most information when labeled [6].

Weak supervision approaches exploit supervision at a higher abstraction or that is noisier in nature to efficiently generate a large number of training examples, even if those are of a lower quality [14,15]. This includes letting non-experts generate labels through crowdsourcing or employing rules and heuristics for labeling data. Multiple weak supervision approaches can be combined to overcome the possibly lower accuracy and coverage of weak supervision [14].

One method of weak supervision is distant supervision [11], where a knowledge base or any other external resource is used to train a supervised algorithm.

While originally applied in the context of relation extraction from text, it has been used for the task of augmenting a knowledge base from semi-structured web data, including web tables [4,10]. Bizer et al. [2] make use of schema.org annotations extracted from 43 thousand e-shops to distantly supervise a deep neural network for product matching. To generate training pairs, they make use of generic product identifies that are often provided along the annotations.

Ratner et al. [15] introduce the data programming paradigm, where any weak supervision strategy, including domain heuristics and distant supervision, can be codified into individual low-coverage labeling functions. The authors focus on denoising noisy and conflicting labels, by assigning accuracies to labeling functions using a generative algorithm. In contrast, we do not label using the individual rules, but first ensemble a set of rules and an unsupervised weighted average rule to create one labeling function per class. While we attempt to overcome the low coverage of our rules using ensembling, the authors do not suggest an approach to overcome the possible low coverage of their labeling functions. Snorkel is a system that enables the use of weak supervision based on the data programming paradigm [14]. Snorkel Drybell adapts Snorkel to exploits diverse organizational knowledge resources. Its effectiveness is evaluated in a large-scale case-study at Google [1].

Snuba [19] is a weak supervision system that uses a small set of labeled data to derive heuristics to generate training data and train a machine learning model. The heuristics are similar in purpose to our rule sets, and the authors also limit themselves to what they term *primitive features*, which in their case are bag-of-words representation for text or bounding box attributes for images. In our case, we limit our self to attribute tests using the schema of a knowledge base. As in our case, training a machine learning model yield an increase in performance, which the authors similarly contribute to the fact that learned models are more expressive and can exploit more features. Snuba still requires hundreds of manually labeled training examples to derive heuristics, whereas in our case experts only need to provide a small number of bold matching rules.

Shen et al. [18] introduce constraint-based entity matching, where they suggest a probabilistic framework within which domain-specific constraints can be exploited to perform entity matching without the need for manual labeling. The introduced constraint are of a broad-variety, and not limited to a specific format. Their work differs from ours, as, first of all, their constraints are generally more complex and not based on simple attribute tests using a predefined schema. This makes providing supervision less straight-forward and possibly more laborious for experts. Additionally, they only provide a matching method that uses the constraints directly, and do not consider using them to bootstrap a supervised machine learning algorithm.

To bootstrap supervised learning, a small number of labeled seed examples are often used [11,13], but there have also been approaches that use alternatives to seeds, e.g. domain-independent patterns [5]. We bootstrap by using a classifier that ensembles a heuristic domain-agnostic matching rule and a limited set of user-provided class-specific matching rule sets.

7 Conclusion

This work investigates the possibility of reducing the effort spent on manually labeling training data for the task of augmenting knowledge bases with long-tail entities from web tables. For this, we introduce and evaluate a weak supervision approach that exploits more efficient supervision at a higher level of abstraction.

Specifically, we suggest, as an alternative to manually labeling thousands of entity matching pairs, the use of a small set of bold user-provided class-specific matching rules. These rules are built upon properties from the schema of a knowledge base class, making them universal and semantically easy to understand. More importantly, these rules require considerably less effort to create. To overcome the possibly limited coverage of these rules, we suggest a method to ensemble these class-specific matching rules with a class-agnostic unsupervised matching model. This yields an effective weakly supervised method for long-tail entity extraction.

We then introduce an approach to bootstrap a supervised learning algorithm by using the weakly supervised method as a labeling function and a set of unlabeled web tables. We find that with bootstrapping, we are able to achieve a performance close to that of supervision with manually labeled data. As a result, we are able to perform long-tail entity extraction with considerably reduced effort spent on supervision.

Our weak supervision approach can be highly useful for a variety of tasks. In case where recall is a secondary objective, our approach can be tuned towards precision and used to add highly accurate, albeit fewer, long-tail entities to a knowledge base. The approach can also be used to facilitate generating training data for manual labeling, where experts must only correct generated labels instead of creating them. This would considerably reduce the effort required for manually labeling training data.

We believe that an interesting direction for future work would be combining weakly supervised labeling functions and active learning. The labeling functions could be used to reduce the effort spent of learning initial models. These models can afterwards be refined by labeling individual examples chosen by the active learning method.

References

1. Bach, S.H., et al.: Snorkel drybell: a case study in deploying weak supervision at industrial scale. In: 2019 International Conference on Management of Data, SIGMOD 2019, pp. 362–375. ACM (2019)
2. Bizer, C., Primpeli, A., Peeters, R.: Using the semantic web as a source of training data. Datenbank-Spektrum **19**(2), 127–135 (2019)
3. Cafarella, M.J., Halevy, A.Y., Zhang, Y., Wang, D.Z., Wu, E.: Uncovering the relational web. In: 11th International Workshop on the Web and Databases, WebDB 2008 (2008)
4. Dong, X., et al.: Knowledge vault: a web-scale approach to probabilistic knowledge fusion. In: 20th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, KDD 2014, pp. 601–610. ACM (2014)

5. Etzioni, O., et al.: Methods for domain-independent information extraction from the web: an experimental comparison. In: Nineteenth National Conference on Artificial Intelligence, AAAI 2004, pp. 391–398 (2004)
6. Han, J., Kamber, M., Pei, J.: Data Mining: Concepts and Techniques, 3rd edn. Morgan Kaufmann, Boston (2012)
7. Hassanzadeh, O., Chiang, F., Lee, H.C., Miller, R.J.: Framework for evaluating clustering algorithms in duplicate detection. Proc. VLDB Endow. **2**(1), 1282–1293 (2009)
8. Hoffart, J., Suchanek, F.M., Berberich, K., Weikum, G.: Yago2: a spatially and temporally enhanced knowledge base from wikipedia. Artif. Intell. **194**, 28–61 (2013)
9. Lehmann, J., et al.: Dbpedia - a large-scale, multilingual knowledge base extracted from wikipedia. Semant. Web **6**(2), 167–195 (2015)
10. Lockard, C., Dong, X.L., Einolghozati, A., Shiralkar, P.: Ceres: distantly supervised relation extraction from the semi-structured web. Proc. VLDB Endow. 11(10) (2018)
11. Mintz, M., Bills, S., Snow, R., Jurafsky, D.: Distant supervision for relation extraction without labeled data. In: Joint Conference of the 47th Annual Meeting of the ACL and the 4th International Joint Conference on Natural Language Processing of the AFNLP. vol. 2, pp. 1003–1011 (2009)
12. Oulabi, Y., Bizer, C.: Extending cross-domain knowledge bases with long tail entities using web table data. In: 22nd International Conference on Extending Database Technology, EDBT 2019, pp. 385–396 (2019)
13. Pennacchiotti, M., Pantel, P.: A bootstrapping algorithm for automatically harvesting semantic relations. In: Fifth International Workshop on Inference in Computational Semantics, ICoS 2006 (2006)
14. Ratner, A., Bach, S.H., Ehrenberg, H., Fries, J., Wu, S., Ré, C.: Snorkel: rapid training data creation with weak supervision. Proc. VLDB Endow. **11**(3), 269–282 (2017)
15. Ratner, A.J., De Sa, C.M., Wu, S., Selsam, D., Ré, C.: Data programming: creating large training sets, quickly. In: Advances in Neural Information Processing Systems, NIPS 2016, vol. 29, pp. 3567–3575 (2016)
16. Ringler, D., Paulheim, H.: One knowledge graph to rule them all? analyzing the differences between dbpedia, yago, wikidata & co. In: KI 2017: Advances in Artificial Intelligence - 40th Annual German Conference on AI, pp. 366–372 (2017)
17. Ritze, D., Bizer, C.: Matching web tables to dbpedia - a feature utility study. In: 20th International Conference on Extending Database Technology, EDBT 2017, pp. 210–221 (2017)
18. Shen, W., Li, X., Doan, A.: Constraint-based entity matching. In: 20th National Conference on Artificial Intelligence, AAAI 2005, vol. 2, pp. 862–867 (2005)
19. Varma, P., Ré, C.: Snuba: automating weak supervision to label training data. Proc. VLDB Endow. **12**(3), 223–236 (2018)
20. Vrandečić, D., Krötzsch, M.: Wikidata: a free collaborative knowledgebase. Commun. ACM **57**(10), 78–85 (2014)